Software-Systeme zielgerichtet bewerten mit LASR

Stefan Zörner, embarc

JUG Saxony Day, Radebeul, 27. September 2024

Abstract

Software-Systeme zielgerichtet bewerten mit LASR

Ein leichtgewichtiger Ansatz für euer Software-Review

Mit Architekturbewertungen ist es möglich, Schwächen und Potenziale von Softwarelösungen herauszuarbeiten, Entscheidungen abzusichern und Verbesserungsmaßnahmen zu bewerten. Klassische Analyseansätze aus diesem Umfeld wie ATAM sind fundiert, kommen aber gerade in beweglichen Softwarevorhaben etwas schwergewichtig, mitunter fast zeremoniell daher.

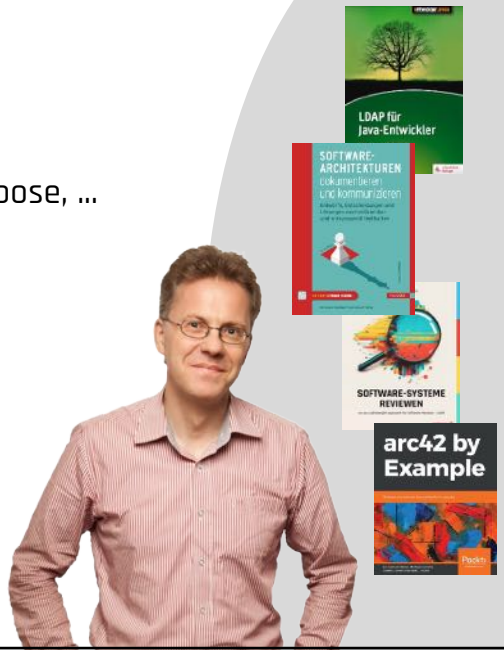
In diesem Vortrag gebe ich einen lebendigen Einstieg in die Welt der Architekturbewertung. Vor allem lernt Ihr eine leichtgewichtige Herangehensweise kennen. Ihr könnt diese mit Eurem Team unmittelbar anwenden, Euer Softwaresystem beleuchten und zügig zu ersten Erkenntnissen kommen. Wir greifen auf die Essenzen etablierter Bewertungsmethoden zurück. Und erarbeiten uns einen roten Faden durch ein Review, inkl. möglicher Vertiefungspunkte für eine höhere Konfidenz im Bewertungsergebnis.

Stefan Zörner

- Softwarearchitekt bei embarc in Hamburg
- Vorher Bayer AG, Mummert + Partner, IBM, oose, …

Schwerpunkte

- Softwarearchitektur (Entwurf, Bewertung, Dokumentation)
- Cloud-Technologien



00. Agenda

Softwarel-Systeme zielgerichtet bewerten mit LASR

01. **Warum leichtgewichtig bewerten?**

02. **Verstehe, was Dich speziell macht**

03. **Durchleuchte die Architektur**

04. **Erhöhe die Konfidenz (bei Bedarf)**

05. **Weitere Informationen**

01.

Warum leichtgewichtig bewerten?

01. Warum leichtgewichtig bewerten?

02. Verstehe, was Dich speziell macht

03. Durchleuchte die Architektur

04. Erhöhe die Konfidenz (bei Bedarf)

05. Weitere Informationen

Was ist Softwarearchitektur?

Softwarearchitektur :=

Σ wichtige Entscheidungen

wichtig =
- fundamental (betrifft viele)
- im weiteren Verlauf nur schwer zu ändern
- entscheidend für den Erfolg Eures Softwaresystems

Themen für Entscheidungen

Zerlegung
Welcher Architekturstil?
Wie zerfällt die Anwendung?
Teilsysteme, Module,
Komponentenbildung,
Abhängigkeiten ...

Zielumgebung
Wo läuft die Software?
Beim Endanwender, im
eigenen Rechenzentrum,
Cloud, Verteilung,
Virtualisierung ...

Technologie-Stack
Was setzen wir ein?
Programmiersprache(n)
Bibliotheken, Frameworks,
Middleware,
Querschnittsthemen ....

Vorgehen
Wie arbeiten wir?
Planen, Entwickeln, Testen,
Bauen, Dokumentieren,
Ausliefern, Nachjustieren, ...

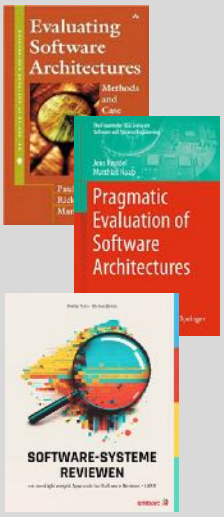
Was ist ein Review?

Wörtlich:
Review == etwas **(über)prüfen**, besprechen, ...

Gegenstand („etwas“) in unserem Fall:
Die Architektur(-entscheidungen) eines Softwaresystems

Typischer Begriff in der Fachwelt / -literatur auch:
Architekturbewertung (englisch „Evaluation“)

Kann durch Außenstehende erfolgen – muss aber nicht.



Kernelemente eines Reviews

Für eine Bewertung oder ein Review braucht es mindestens

Einen Anlass
Warum bewerten wir?

Einen Gegenstand
Was bewerten wir?

Einen Maßstab
Wonach (wo gegen) bewerten wir?

Kernelemente eines Reviews

Für eine Bewertung oder ein Review braucht es mindestens

Einen Anlass
Warum bewerten wir?

Einen Gegenstand
Was bewerten wir?

Einen Maßstab
Wonach (wo gegen) bewerten wir?

Typische Anlässe (… findet Ihr Euch wieder?)

Anlass 1 („Neuanfang“)

Eine **Neuentwicklung** steht an und
erste Lösungsansätze stehen im Raum.

Leitfrage:
Seid Ihr und Euer Team auf dem **richtigen Weg**?

Typische Anlässe (… findet Ihr Euch wieder?)

Anlass 2 („Wünsche“)

Unterschiedliche Stakeholder verfolgen
widersprüchliche Ziele mit Eurer Software.

Leitfrage:
Wie **konkretisiert** und **priorisiert** Ihr deren Wünsche?

Typische Anlässe (... findet Ihr Euch wieder?)

Anlass 3 („Unruhe“)

Das **Management** hat das **Vertrauen** in Eure Lösung verloren.

Leitfrage:
Wie gewinnt Ihr es zurück und strahlt **Sicherheit** aus?

embarc.de
LASR. Ein leichtgewichtiger Ansatz

Typische Anlässe (... findet Ihr Euch wieder?)

Anlass 4 („Legacy“)

Größere **Umbaumaßnahmen** in Eurer Software stehen an.

Leitfrage:
Wie wählt Ihr passende Lösungsansätze **nachvollziehbar** aus?

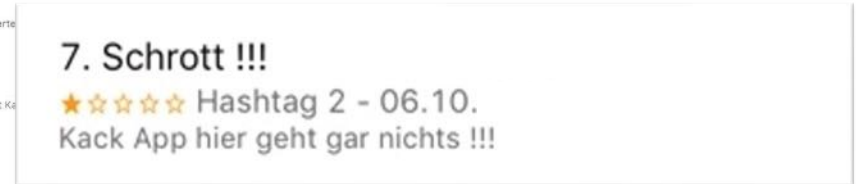
Das Leistungsversprechen von Reviews

In all diesen Situationen unterstützen Review-Ansätze und helfen Euch und Eurem Team die Leitfragen zu beantworten.

Konkret: Software-Reviews …

- … **decken** Kompromisse und **Risiken** von Softwarelösungen **auf**.
- … **sichern** technische und architektonische **Ideen ab**.

Nörgeln ist einfach …



Bewertungsmethoden (Auswahl)

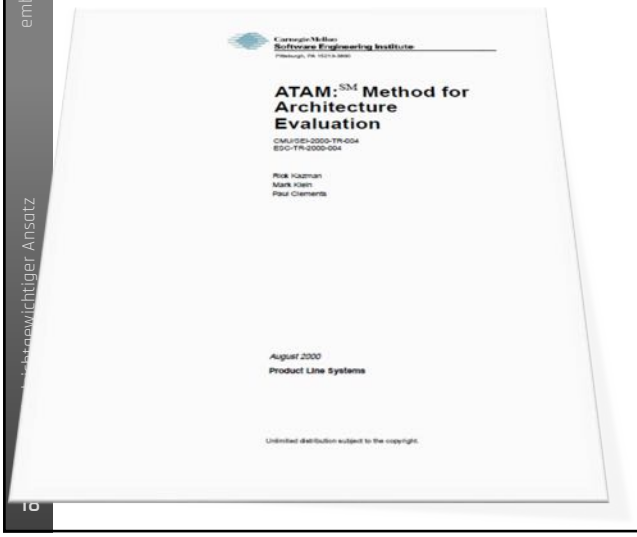
SAAM Software Architecture Analysis Method	**ARID** Architecture Review for Intermediate Designs
ATAM Architecture Tradeoff Analysis Method	**DCAR** Decision Centric Architecture Review
CBAM Cost-Benefit Analysis Method	**DASE** Decision and Scenario based architecture evaluation
TARA Tiny Architecture Review Approach	**Pre-Mortem** Risk-Brainstorming and Mitigation
PBAR Pattern Based Architecture Review	**LASR** Lightweight Approach for Software Reviews

embarc.de
LASR. Ein leichtgewichtiger Ansatz
17

Bewertungsmethoden (Auswahl)

SAAM Software Architecture Analysis Method	**ARID** Architecture Review for Intermediate Designs
ATAM Architecture Tradeoff Analysis Method	**DCAR** Decision Centric Architecture Review
CBAM Cost-Benefit Analysis Method	**DASE** Decision and Scenario based architecture evaluation
TARA Tiny Architecture Review Approach	**Pre-Mortem** Risk-Brainstorming and Mitigation
PBAR Pattern Based Architecture Review	**LASR** Lightweight Approach for Software Reviews

ATAM



Architecture Tradeoff Analysis Method

- **Akademischer Ursprung**: Carnegie Mellon University, 2000
- Bekannteste Methode zur Bewertung von Softwarearchitektur
- **Qualitativer** Ansatz, Szenarien- und **Workshop**-basiert
- Früh anwendbar, geht von den **Zielen** aus

Herausforderungen …

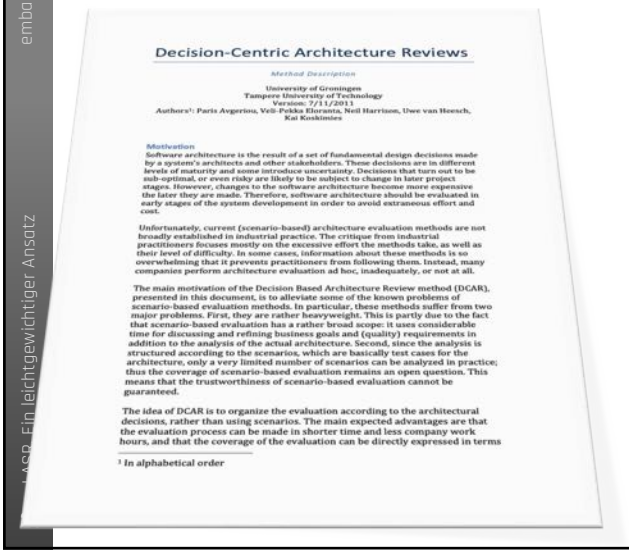
Die Anwendung fundierter Bewertungsmethoden ist mitunter schwierig.

- Der Einsatz erfordert häufig **viele Beteiligte**.
- Es sind oftmals **Vorarbeiten nötig**, beispielsweise die Aufbereitung der Geschäftsziele und das Anfertigen eines Architekturüberblicks.
- Sie liefern **nur Rohergebnisse**, die für eine effiziente Kommunikation aufwendig nachzubearbeiten sind.
- Durchführung und Moderation verlangen einiges ab – die Methoden **unterstützen** dabei **wenig**.

Bewertungsmethoden (Auswahl)

SAAM Software Architecture Analysis Method	**ARID** Architecture Review for Intermediate Designs
ATAM Architecture Tradeoff Analysis Method	**DCAR** Decision Centric Architecture Review
CBAM Cost-Benefit Analysis Method	**DASE** Decision and Scenario based architecture evaluation
TARA Tiny Architecture Review Approach	**Pre-Mortem** Risk-Brainstorming and Mitigation
PBAR Pattern Based Architecture Review	**LASR** Lightweight Approach for Software Reviews

DCAR



Decision-Centric Architecture Reviews

Method Description

University of Groningen
Tampere University of Technology
Version: 7/11/2011
Authors[1]: Paris Avgeriou, Veli-Pekka Eloranta, Neil Harrison, Uwe van Heesch, Kai Koskimies

Motivation

Software architecture is the result of a set of fundamental design decisions made by a system's architects and other stakeholders. These decisions are in different levels of maturity and some introduce uncertainty. Decisions that turn out to be sub-optimal, or even risky are likely to be subject to change in later project stages. However, changes to the software architecture become more expensive the later they are made. Therefore, software architecture should be evaluated in early stages of the system development in order to avoid extraneous effort and cost.

Unfortunately, current (scenario-based) architecture evaluation methods are not broadly established in industrial practice. The critique from industrial practitioners focuses mostly on the excessive effort the methods take, as well as their level of difficulty. In some cases, information about these methods is so overwhelming that it prevents practitioners from following them. Instead, many companies perform architecture evaluation ad hoc, inadequately, or not at all.

The main motivation of the Decision Based Architecture Review method (DCAR), presented in this document, is to alleviate some of the known problems of scenario-based evaluation methods. In particular, these methods suffer from two major problems. First, they are rather heavyweight. This is partly due to the fact that scenario-based evaluation has a rather broad scope: it uses considerable time for discussing and refining business goals and (quality) requirements in addition to the analysis of the actual architecture. Second, since the analysis is structured according to the scenarios, which are basically test cases for the architecture, only a very limited number of scenarios can be analyzed in practice; thus the coverage of scenario-based evaluation remains an open question. This means that the trustworthiness of scenario-based evaluation cannot be guaranteed.

The idea of DCAR is to organize the evaluation according to the architectural decisions, rather than using scenarios. The main expected advantages are that the evaluation process can be made in shorter time and less company work hours, and that the coverage of the evaluation can be directly expressed in terms

[1] In alphabetical order

Decision-Centric Architecture Review

- Ursprung: Universitäten von Groningen (NL), Tampere (FI), 2011
- Initial ähnlicher Ablauf zu ATAM, produziert auch ähnliche Ergebnisse
- Geht in der Analyse von den (Architektur-)Entscheidungen aus

Bewertungsmethoden (Auswahl)

SAAM Software Architecture Analysis Method	**ARID** Architecture Review for Intermediate Designs
ATAM Architecture Tradeoff Analysis Method	**DCAR** Decision Centric Architecture Review
CBAM Cost-Benefit Analysis Method	**DASE** Decision and Scenario based architecture evaluation
TARA Tiny Architecture Review Approach	**Pre-Mortem** Risk-Brainstorming and Mitigation
PBAR Pattern Based Architecture Review	**LASR** Lightweight Approach for Software Reviews

Was ist leichtgewichtig?

Merkmale eines leichtgewichtigen Reviews

- Die **Anzahl** der Beteiligten / **Stakeholder** ist **gering**.
- **Aufwand** und Dauer sind **überschaubar**.
- **Ergebnisse** liegen vergleichsweise **schnell** vor.

Konkret z.B.

- Entwicklungsteam führt Review **allein** und **ohne Vorbereitung** durch.
- Bereits nach einem halben Tag liegt ein **kommunizierbares Ergebnis** vor.

Erfahrungswissen

Software-Systeme reviewen
mit dem Lightweight Approach
for Software Reviews - LASR

Leanpub

Autoren: Stefan Toth, Stefan Zörner
Verlag: Leanpub, September 2023
Sprache: Deutsch, EPUB, PDF

➔ leanpub.com/software-systeme-reviewen/

Markante Merkmale von LASR

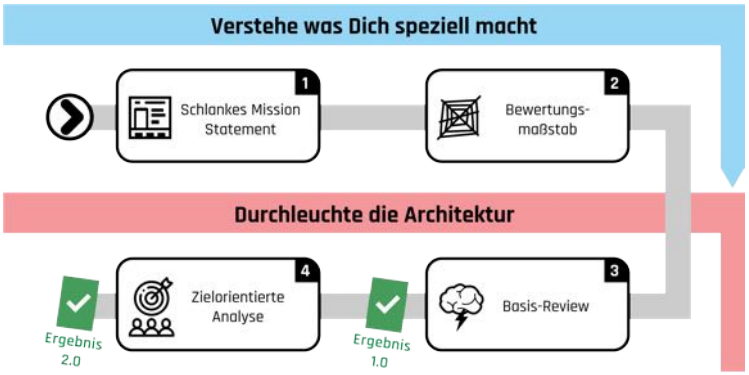
- Schlankere Methode als ATAM, trotzdem **zielorientiert**
- Mit dem **eigenen Team** und potentiell **alleine durchführbar**
- Liefert **schnell** ein **erstes Ergebnis**, z.B. an einem Nachmittag
- **Spinnennetzgraphik** zur Erkenntnisverdichtung und -kommunikation
- optionale Aktivitäten zur schrittweisen **Konfidenzerhöhung bei Bedarf**
- **Unterstützungsmaterial** für Bewertungs-maßstab, Risikofindung und Ergebniserarbeitung

Ein schlanker Ansatz

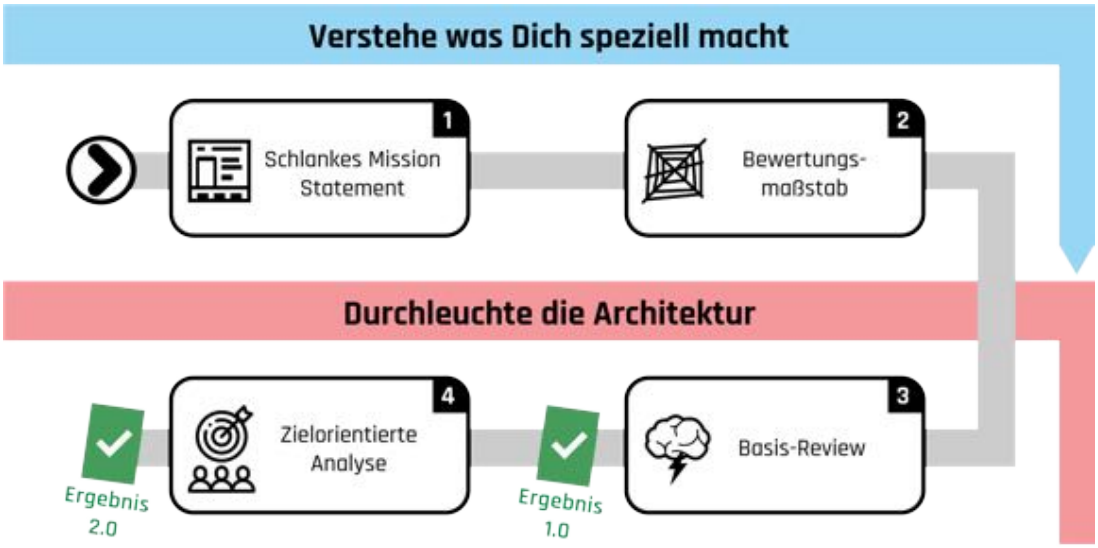
LASR (Lightweight Approach for Software-Reviews) ist ein strukturiertes Vorgehen für leichtgewichtige Software-Reviews.

LASR Kern-Review

- 2 Tätigkeiten
- 4 Schritte
- Ein frühes erstes Ergebnis



Tätigkeiten und Schritte im Kern-Review



02.

Verstehe, was Dich speziell macht

01. Warum leichtgewichtig bewerten?

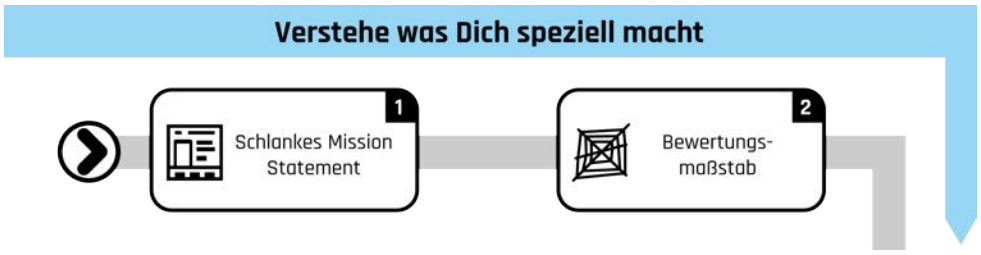
02. Verstehe, was Dich speziell macht

03. Durchleuchte die Architektur

04. Erhöhe die Konfidenz (bei Bedarf)

05. Weitere Informationen

Verstehe, was Dich speziell macht



Was ist zu tun?

- Die Aufgabenstellung für das System zu einem prägnanten Mission Statement verdichten
- Die Top 3-5 Ziele des Systems identifizieren
- Jeweiliges Ziel-Level festlegen

Kernelemente eines Reviews

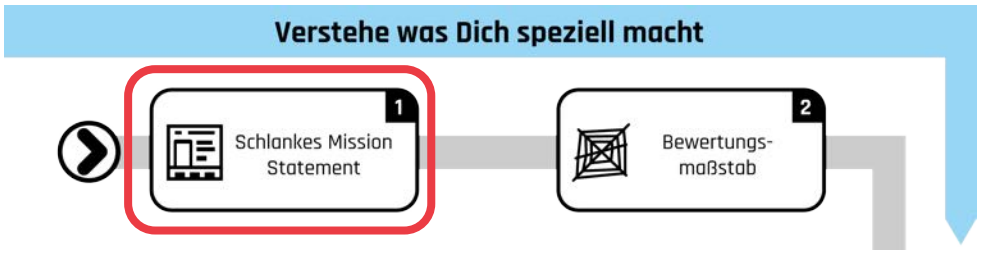
Für eine Bewertung oder ein Review braucht es mindestens

Einen Anlass
Warum bewerten wir?

Einen Gegenstand
Was bewerten wir?

Einen Maßstab
Wonach (wo gegen) bewerten wir?

Verstehe, was Dich speziell macht



Was ist zu tun?

- **Die Aufgabenstellung für das System zu einem prägnanten Mission Statement verdichten**
- Die Top 3-5 Ziele des Systems identifizieren
- Jeweiliges Ziel-Level festlegen

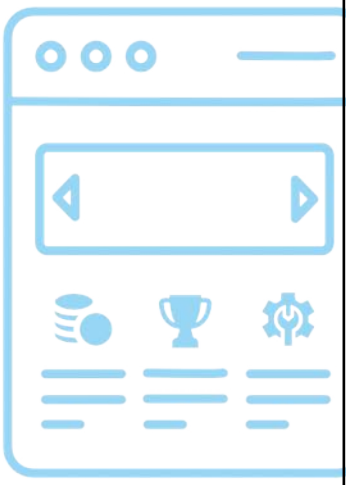
Mission Statement

Der erste Schritt des Reviews **macht** den **Bewertungsgegenstand** mit Hilfe eines schlanken Mission Statements **explizit**.

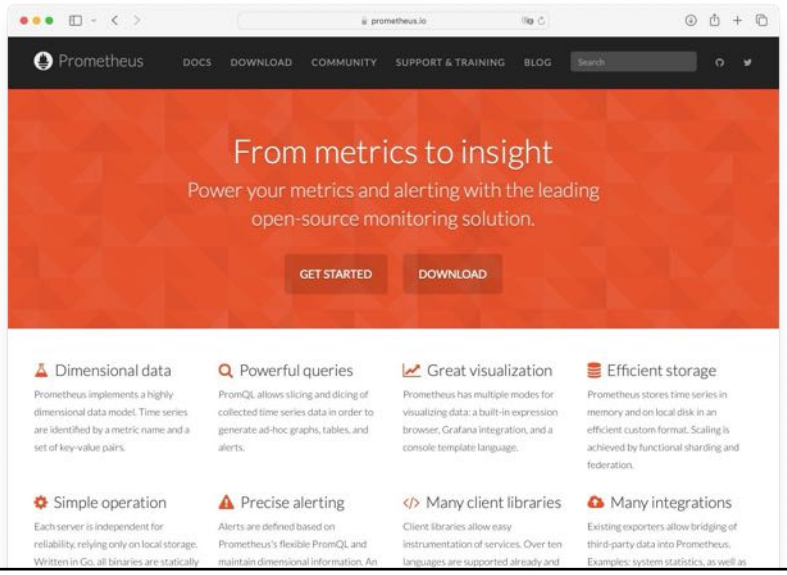
Landing-Page als Metapher

Ziel: Eine prägnante und kraftvolle Beschreibung des Systemzwecks.

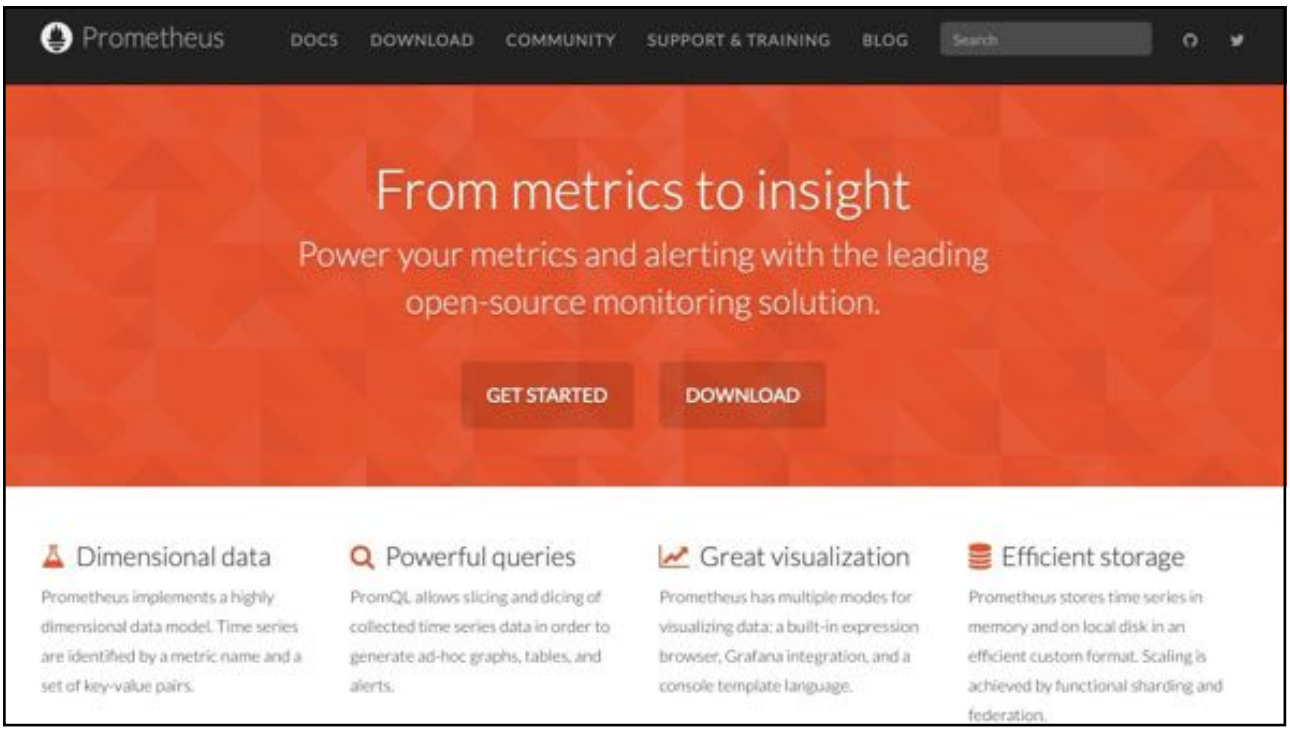
Die Landing-Pages prominenter Software-Produkte bieten **Orientierung** bei der Erstellung.



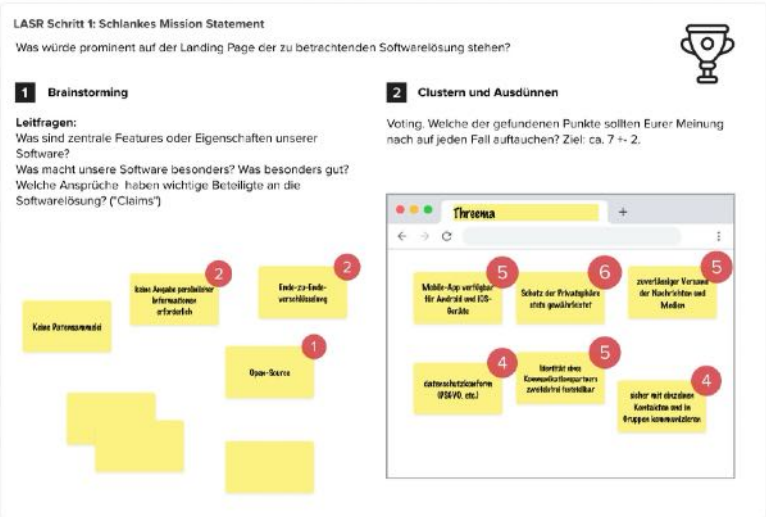
Beispiel: Landing Page von Prometheus



→ prometheus.io



Erarbeiten im Workshop



Beispiel im digitalen Whiteboard (Mural)

Quelle: S. Toth, S. Zörner: „Software-Systeme reviewen", Leanpub 2023

Qualitätsmerkmale (Begriffe nach ISO 25010:2023)

Funktionale Eignung
(Functional Suitability)
Sind die berechneten Ergebnisse genau genug / exakt, ist die Funktionalität angemessen? ...

Effizienz
(Performance Efficiency)
Antwortet die Software schnell, hat sie einen hohen Durchsatz, einen geringen Ressourcenverbrauch? ...

Kompatibilität
Compatibility)
Ist die Software konform zu Standards, arbeitet sie gut mit anderen zusammen? ...

Benutzbarkeit
(Interaction Capability)
Ist die Software intuitiv zu bedienen, wiedererkennbar, leicht zu erlernen, attraktiv? ...

Zuverlässigkeit
(Reliability)
Ist das System verfügbar, tolerant gegenüber Fehlern, nach Abstürzen schnell wieder hergestellt? ...

Sicherheit
(Security)
Ist das System sicher vor Angriffen? Sind Daten und Funktion vor unberechtigtem Zugriff geschützt? ...

Wartbarkeit
(Maintainability)
Ist die Software leicht zu ändern, erweitern, testen, verstehen? Lassen sich Teile wiederverwenden? ...

Übertragbarkeit
(Flexibility)
Ist die Software leicht auf andere Situationen oder Zielumgebungen (z.B. anderes OS) übertragbar? ...

Betriebssicherheit
(Safety)
Sind Personen, Tiere, Sachen und Umwelt vor Schäden durch die Software geschützt? ...

Kernelemente eines Reviews

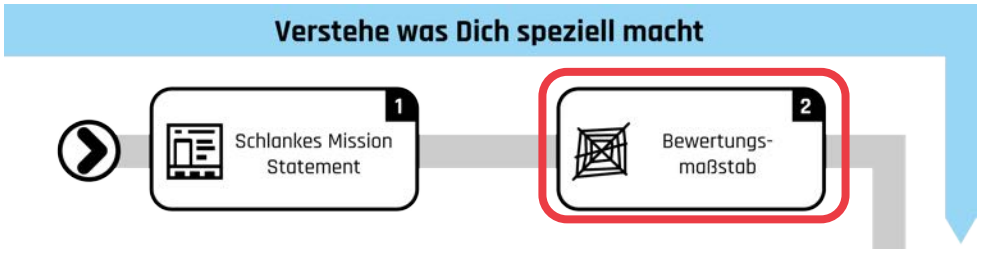
Für eine Bewertung oder ein Review braucht es mindestens

Einen Anlass
Warum bewerten wir?

Einen Gegenstand
Was bewerten wir?

Einen Maßstab
Wonach (wo gegen) bewerten wir?

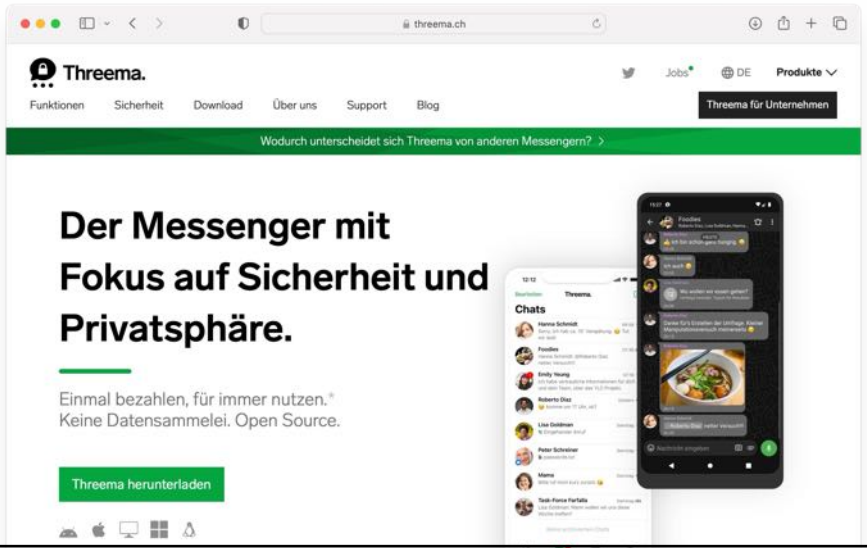
Verstehe, was Dich speziell macht



Was ist zu tun?

- Die Aufgabenstellung für das System zu einem prägnanten Mission Statement verdichten
- **Die Top 3-5 Ziele des Systems identifizieren**
- Jeweiliges Ziel-Level festlegen

Weiteres Beispiel für eine Startseite



Die Qualitätsziele von Threema

Sicherheit	Kommunikations- und Nutzerdaten sind geschützt.
Benutzbarkeit	Einfach zu verwenden.
Zuverlässigkeit	Zuverlässig und effizient im Betrieb.
Kompatibilität	Interoperabel über alle Plattformen hinweg.
Wartbarkeit	Leicht erweiterbar und anpassbar.

Quelle: „Architektur-Porträt: Threema“, Stefan Zörner 2023

Top-5-Challenger (Unterstützungsmaterial)

Material:

- 14 Karten mit Qualitätsmerkmalen

Vorbereitung:

- Mischt die Karten in einem verdeckten Stapel.
- Deckt 5 Karten zufällig auf und legt sie nebeneinander. (das ist Eure Start Top-5)
- Legt die übrigen 9 in einer 3 x 3 Matrix für alle gut sichtbar offen aus.

Top-5-Challenger, Vorbereitung (Beispiel)

Start Top-5

Kandidaten

Ablauf einer Runde

Einen „Challenger“ nominieren

- Team-Mitglied schlägt Ziel vor, das in Top-5 fehlt.
- Die Gruppe diskutiert darüber.
- Der Challenger verdrängt eine Karte oder landet selbst auf dem Ablagestapel.

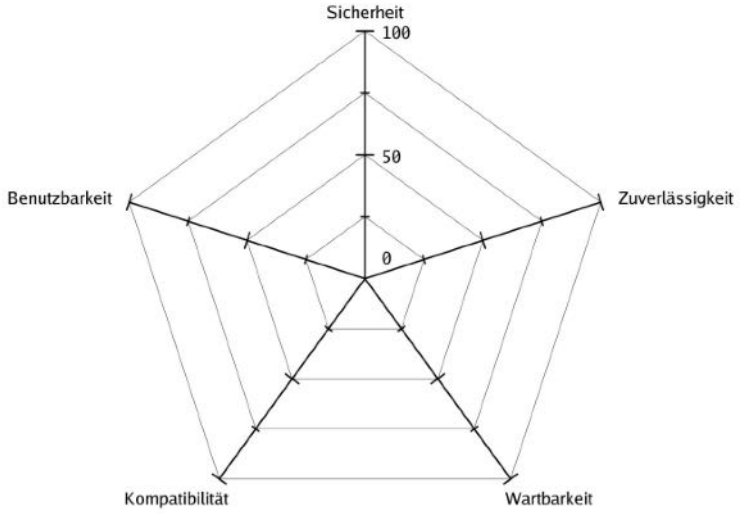
Nach 2 Runden, Beispiel (Beispiel)

Aktuelle Top-5

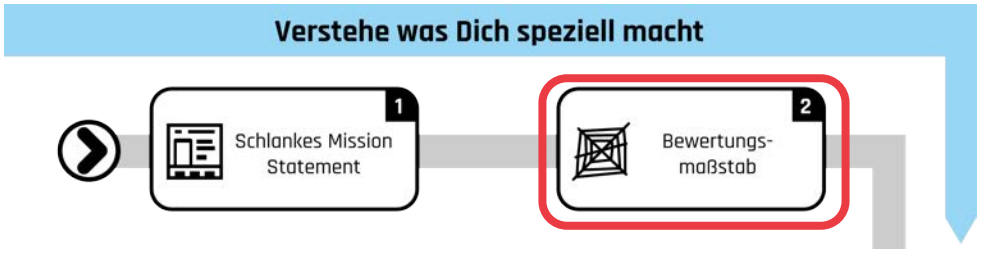
Ablagestapel

Kandidaten

Ergebnis nach diesem Schritt (Beispiel)



Verstehe, was Dich speziell macht

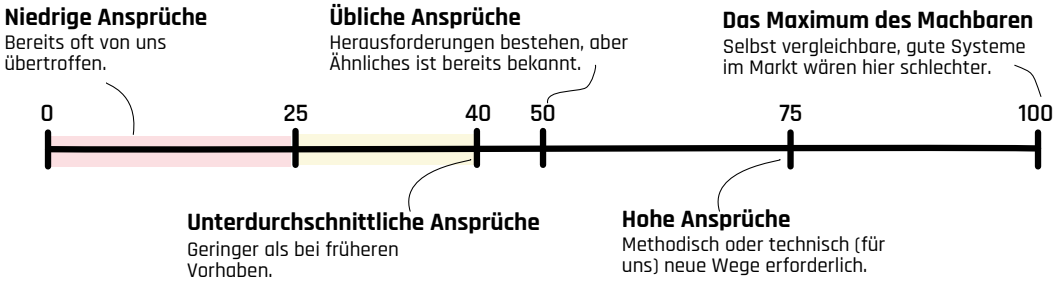


Was ist zu tun?

- Die Aufgabenstellung für das System zu einem prägnanten Mission Statement verdichten
- Die Top 3-5 Ziele des Systems identifizieren
- **Jeweiliges Ziel-Level festlegen**

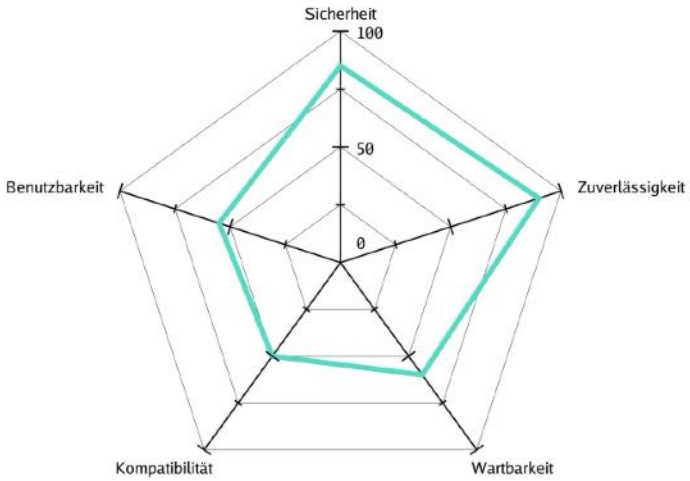
Zielwerte bestimmen

Der Zielwert zwischen 0 und 100 beschreibt jeweils, wie hoch die Erwartungen in dem Bereich sind. Im Vergleich zu anderen Zielen der Top-3-5 und anderen Vorhaben aus dem Kontext des Unternehmens / der Organisation.



Hinweis: Es handelt sich hier um eine Einschätzung, nichts Messbares.

Bewertungsmaßstab einzeichnen



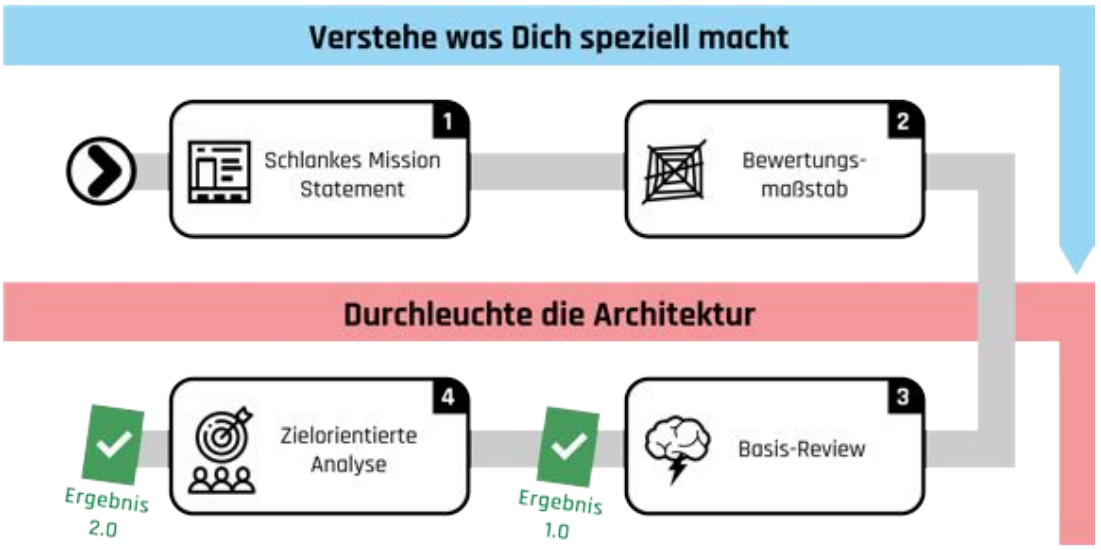
LASR-Ergebnisdiagramm

Die Top-3-5 Qualitätsziele bilden die Achsen des Diagramms.

Die Zielwerte spannen darauf eine grüne Linie auf.

Beispiel

Mit Schritt 2 steht der Ergebnisrahmen.

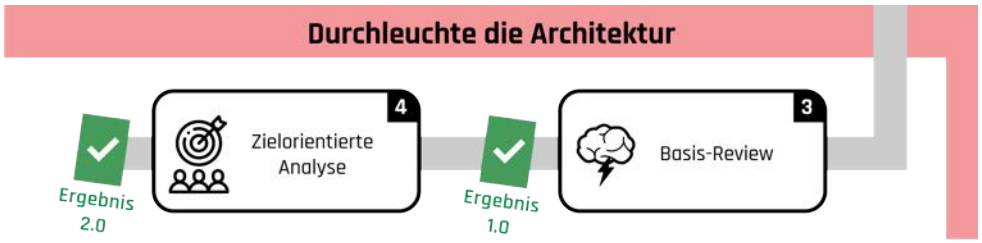


03.

Durchleuchte die Architektur

01. Warum leichtgewichtig bewerten?
02. Verstehe, was Dich speziell macht
03. Durchleuchte die Architektur
04. Erhöhe die Konfidenz (bei Bedarf)
05. Weitere Informationen

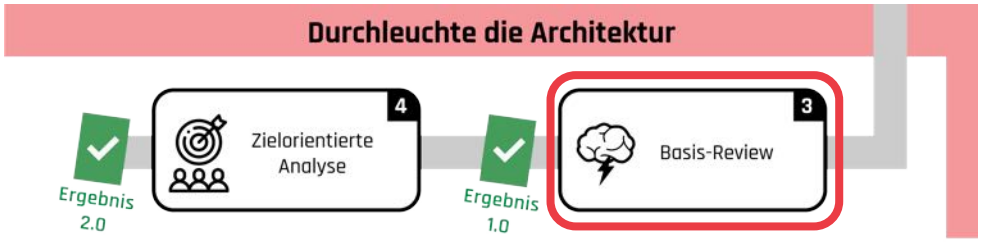
Durchleuchte die Architektur



Was ist zu tun?

- Die größten Risiken des Systems identifizieren
- Die aus den Risiken resultierende Abweichung zu den Zielen quantifizieren („Lücken“)
- Zielorientierte Analysen durchführen, um das Review-Ergebnis bei Bedarf zu schärfen

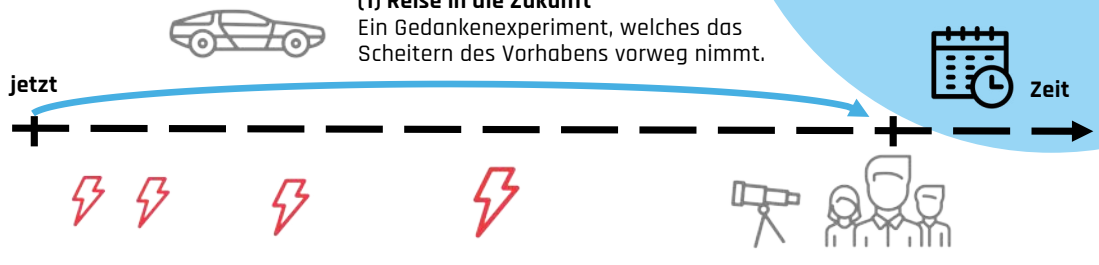
Durchleuchte die Architektur



Was ist zu tun?

- **Die größten Risiken des Systems identifizieren**
- Die aus den Risiken resultierende Abweichung zu den Zielen quantifizieren („Lücken“)
- Zielorientierte Analysen durchführen, um das Review-Ergebnis bei Bedarf zu schärfen

Was ist ein Pre-Mortem?



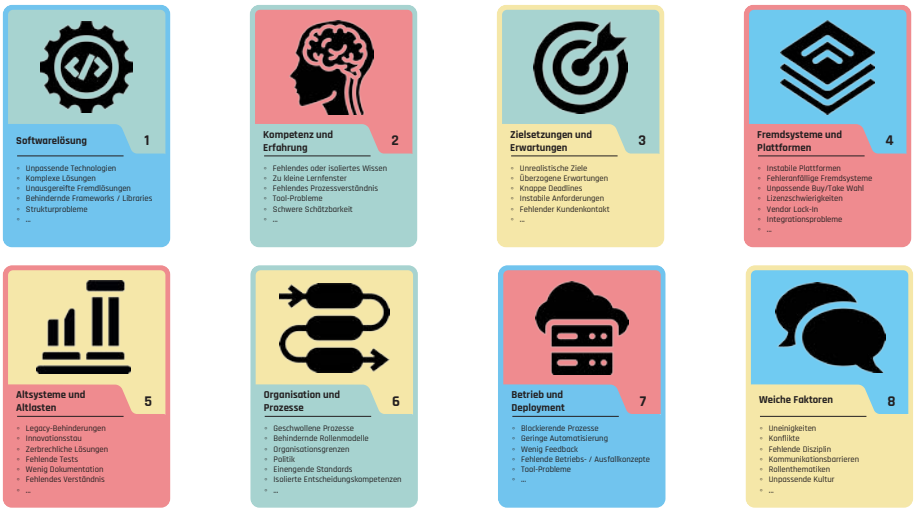
(1) Reise in die Zukunft
Ein Gedankenexperiment, welches das Scheitern des Vorhabens vorweg nimmt.

(2) Blick zurück
Das Team blickt gemeinsam zurück und überlegt, warum das Vorhaben gescheitert ist.

(3) Gründe des Scheiterns/Risiken
Probleme unterschiedlichster Art und Größe machen das Scheitern „plausibel".

- Pre-Mortems werfen einen kritischen Blick auf Vorhaben. Und zwar aus einer hypothetischen Zukunft, in der das Vorhaben gescheitert ist.
- Aus dieser Perspektive ergeben sich „Gründe des Scheiterns", die aus heutiger Sicht mögliche Risiken darstellen.

Die 8 Risiko-Kategorien von LASR



Typische Risikothemen als Ideengeber

Brainstorming-Unterstützung: Das LASR-Kartenset hält insgesamt 32 konkrete Risikokarten als Ideengeber parat, 4 in jeder der 8 Kategorien.

Disclaimer

- Die folgenden **Beispiel-Risiken** sind **realistisch** für Dinge, die Team-Mitglieder im Rahmen eines Risiko-Workshops im Pre-Mortem aufwerfen.
- Bezogen auf unser Beispiel **Threema** hier sind sie **rein fiktiv**. Sie dienen der Illustration.

Beispielkarte

embarc.de

LASR. Ein leichtgewichtiger Ansatz

60

Hypothetisches Risiko (1)

Aufhebung des Gruppen-Limits

„Wir haben die Begrenzung in Gruppen-Größen aufgehoben. Unsere User sind begeistert von den neuen Möglichkeiten. Später bricht der Chat-Server im Threema-Backend unter der massiven Last und der Menge zu speichernden Nachrichten zusammen.“

embarc.de

LASR. Ein leichtgewichtiger Ansatz

61

Hypothetisches Risiko (2)

Das Electron-Framework fällt weg

„Wir waren gezwungen das Electron-Framework in unserer Desktop-App durch eine Alternative / etwas Selbstgebautes zu ersetzen. Das zieht sich. Zwischenzeitlich haben wir den Download des Desktop-Clients von der Seite genommen, weil er auf aktuellen OS nicht mehr funktioniert. “

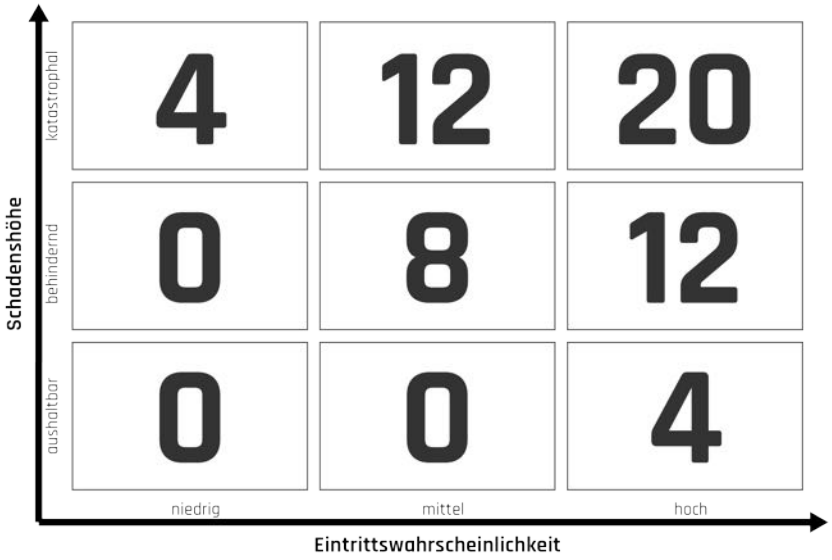
Hypothetisches Risiko (3)

Private Kommunikation scannen

„Die europäische Gesetzgebung hat Messenger-Anbieter verpflichtet, sämtliche private Kommunikation und Dateien zu durchleuchten. Threema muss daraufhin die Lösung für private Nutzer komplett neu denken. “

Risiken einordnen

?



Den betroffenen Zielen zuordnen

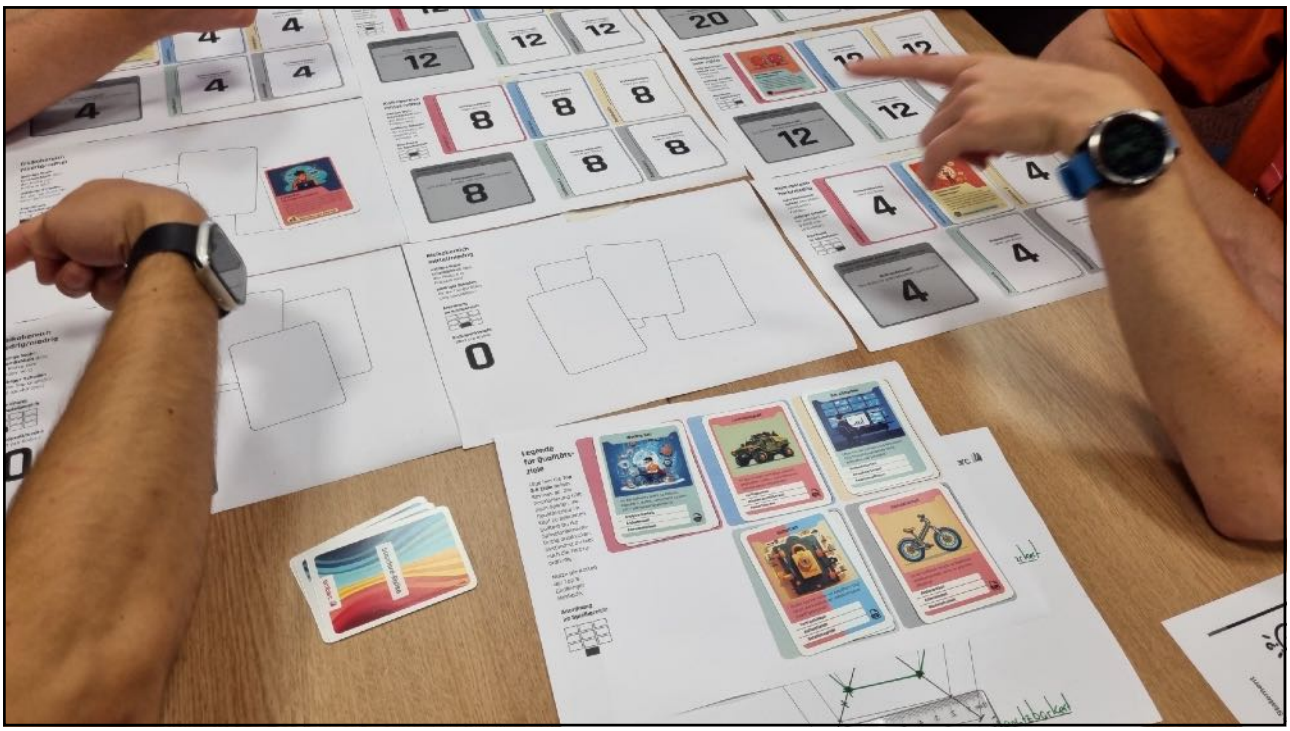
zoom in

Risikobereich mittel/hoch

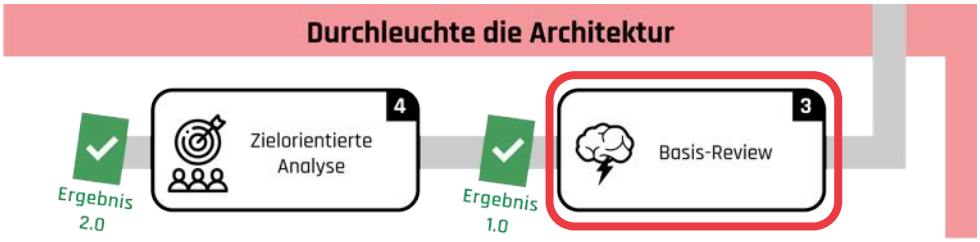
mittlere Wahrscheinlichkeit dass das Risiko zum Problem wird

hoher Schaden für zumindest ein Qualitätsziel (katastrophal)

Anordnung im Spielbereich:



Durchleuchte die Architektur



Was ist zu tun?

- Die größten Risiken des Systems identifizieren
- **Die aus den Risiken resultierende Abweichung zu den Zielen quantifizieren („Lücken“)**
- Zielorientierte Analysen durchführen, um das Review-Ergebnis bei Bedarf zu schärfen

embarc

embarc.de

LASR. Ein leichtgewichtiger Ansatz

Von Risiken zu „Lücken“

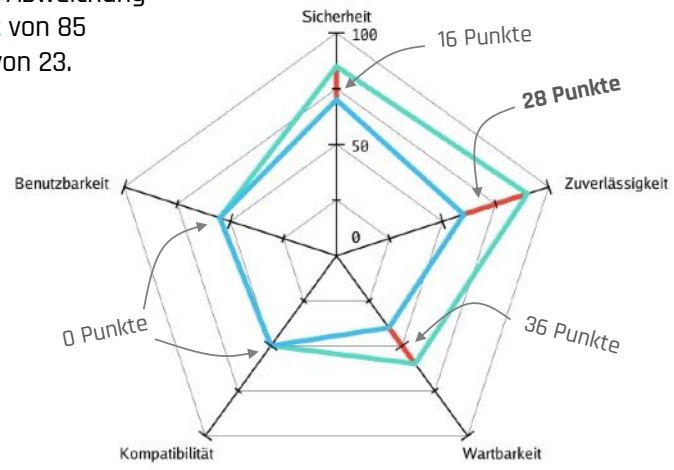
28% Abweichung
vom **Ziellevel**
auf der **Qualitätszielachse**

embarc.de

LASR. Ein leichtgewichtiger Ansatz

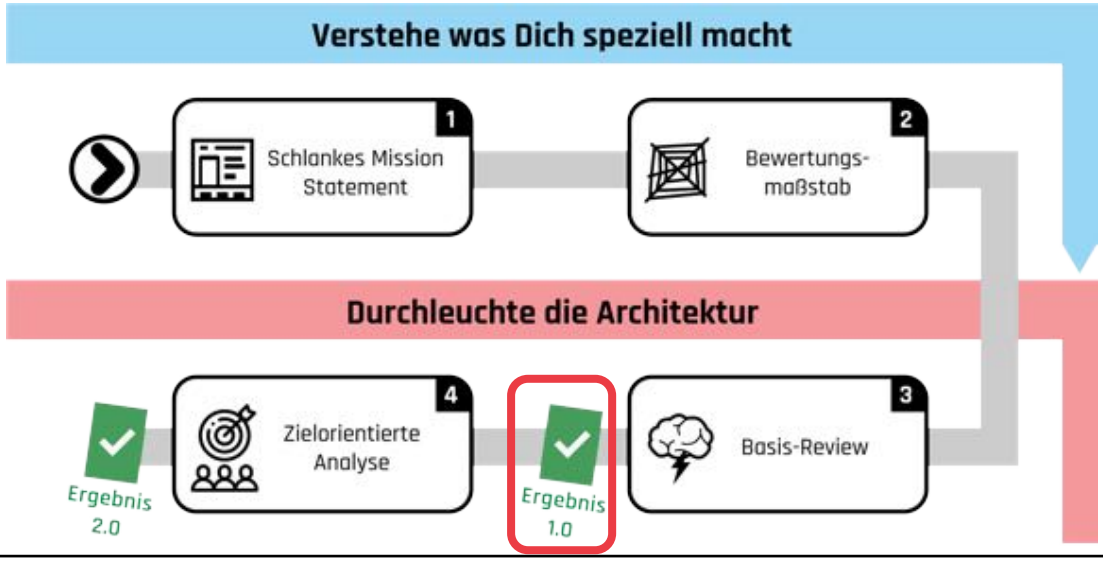
Abweichungen einzeichnen

Beispiel
28 Prozent-Punkte Abweichung
bei einem **Zielwert** von 85
ergibt eine **Lücke** von 23.

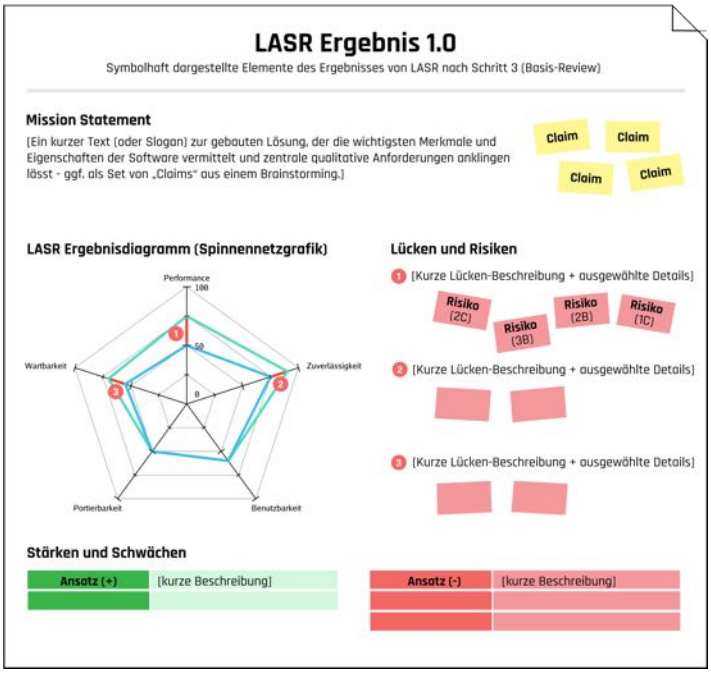


Ziellinie
Abweichungen (Lücken)
Ergebnislinie

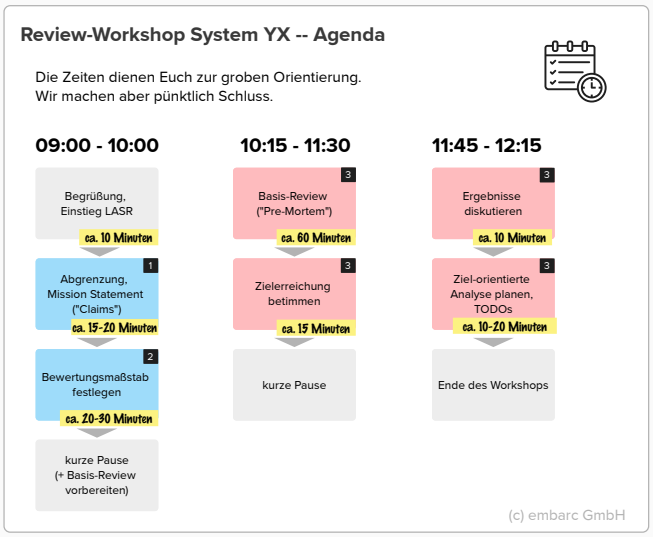
Basis-Review, 1. Ergebnis nach Schritt 3



Ergebnis 1.0



LASR an einem Vormittag



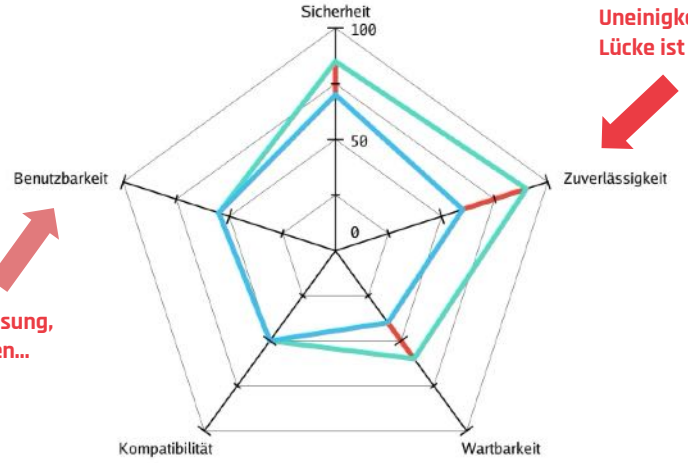
Beispiel-Agenda

Quelle: S. Toth, S. Zörner: „Leichtgewichtige Software-Reviews mit LASR ", Informatik Aktuell 2024

04. Erhöhe die Konfidenz (bei Bedarf)

01. Warum leichtgewichtig bewerten?
02. Verstehe, was Dich speziell macht
03. Durchleuchte die Architektur
04. Erhöhe die Konfidenz (bei Bedarf)
05. Weitere Informationen

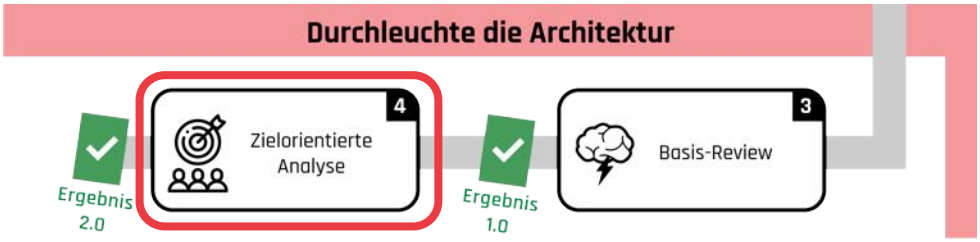
Fokus: Zielorientierte Analyse (Beispiele)



Uneinigkeit!
Lücke ist strittig

Keine Stärken in der Lösung,
Kein explizites Vorgehen...
Trotzdem keine Lücke?

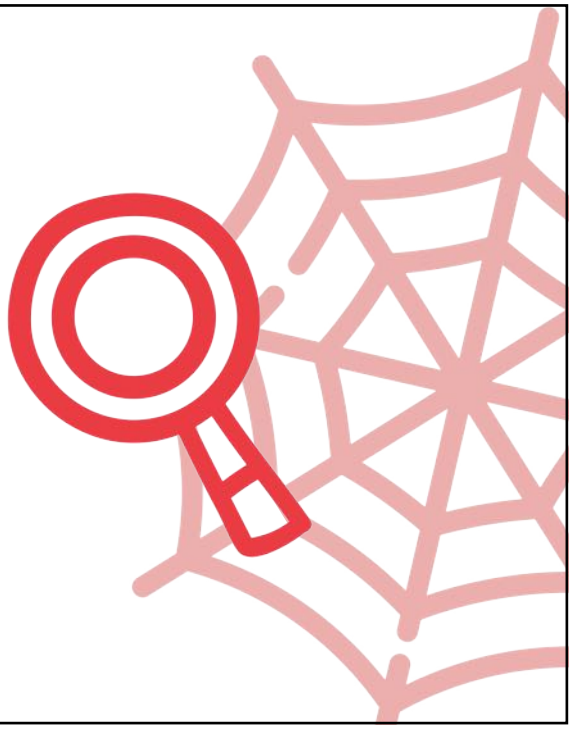
Durchleuchte die Architektur



Was ist zu tun?

- Die größten Risiken des Systems identifizieren
- Die aus den Risiken resultierende Abweichung zu den Zielen quantifizieren („Lücken“)
- **Zielorientierte Analysen durchführen, um das Review-Ergebnis bei Bedarf zu schärfen**

Zielorientierte Analyse



Anhand wichtiger
Qualitätsaussagen:

- Architekturansätze finden
- Kompromisse aufdecken
- Weitere Risiken finden

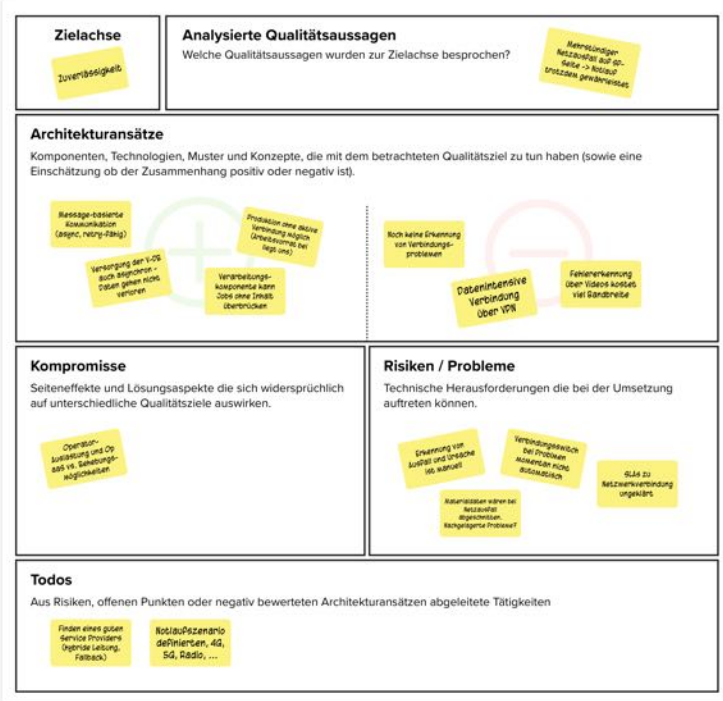
Anschließend ggf. auf
Lösungsebene tiefer graben:

- „Code-Stöbern“
- Vorhandene Messungen nutzen

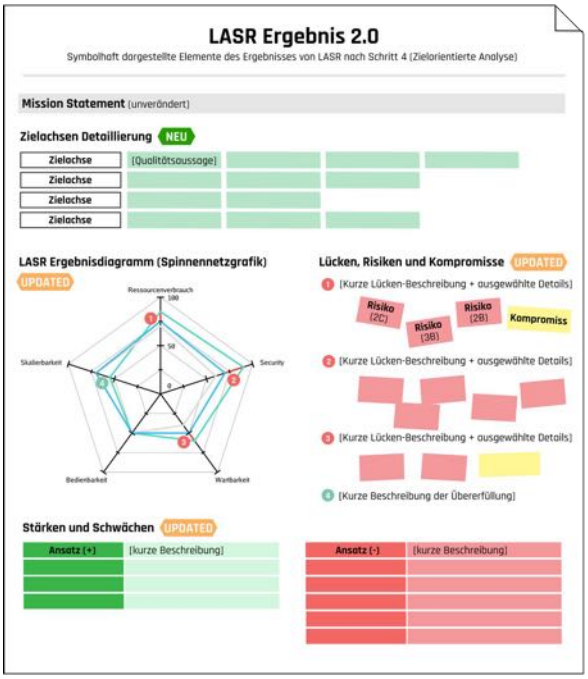
Template

Vorlage, um die Ergebnisse der Analyse zu sammeln, zu verdichten und TODOs abzuleiten.

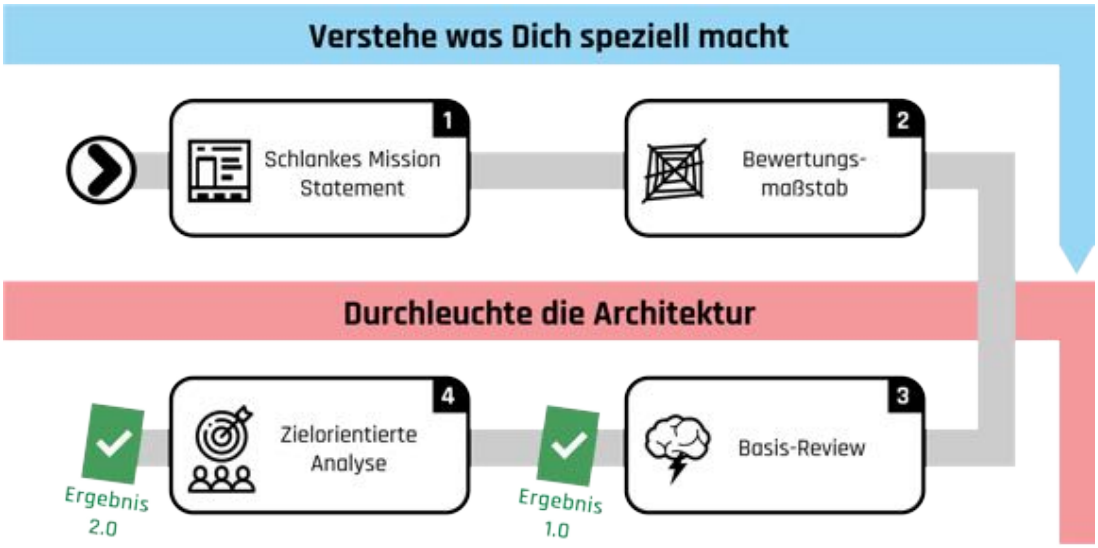
Beispiel



Ergebnis 2.0



Tätigkeiten und Schritte im Kern-Review



Typische Unsicherheiten im Anschluss

Hängt da noch mehr dran?
Gefundene **Risiken** sind in ihrer Natur oder Auswirkung **zu wenig verstanden**, um direkt hilfreiche Aktivitäten daraus abzuleiten.

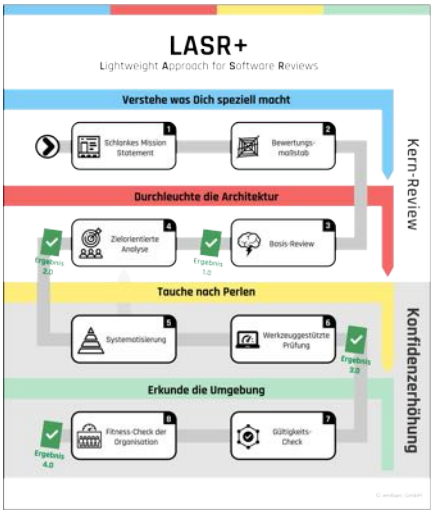
In der Theorie OK, aber was ist mit …?
Rahmenbedingungen (z.B. Standards) oder die eigene **Organisation** wurde als Risikofaktor **zu wenig betrachtet**.

Wo ist der Code?
Die **Architekturebene** wird als **zu abstrakt** für die tatsächlichen Probleme des Vorhabens wahrgenommen.

Wo fangen wir an?
Die **Priorisierung** der identifizierten Probleme ist **schwierig**, ggf. fehlt es bei kleinteiligen Ergebnissen an Überblick oder Einordnung.

Ist nicht auch XY wichtig?
Der **Fokus** auf max. 5 Qualitätsziele wirkt **problematisch.**

Ausblick: Nach dem Kern-Review …



LASR gliedert sich grob in zwei Modi, von denen der zweite optional ist („LASR+“):

I. Kern-Review

Liefert rasch ein erstes Review-Ergebnis, das im vierten Schritt bereits geschärft wird.

II. Konfidenzerhöhung (optional)

Bietet bei Unsicherheiten mit dem Ergebnis mehr Tiefe und schließt Code-Ebene und Organisationsseite mit ein.

05. Weitere Informationen

01. Warum leichtgewichtig bewerten?

02. Verstehe, was Dich speziell macht

03. Durchleuchte die Architektur

04. Erhöhe die Konfidenz (bei Bedarf)

05. Weitere Informationen

TL;DR – Too long; didn't read.

Reviews decken Schwächen von Softwarelösungen auf und **sichern** technische und architektonische Ideen **ab**. **Konventionelle** Bewertungsmethoden wie ATAM sind **mitunter** zu **schwergewichtig**.

Ein **leichtgewichtiges Review** zeichnet sich durch geringere Laufzeit und Aufwände, weniger Beteiligte und **raschere Ergebnisse** aus.

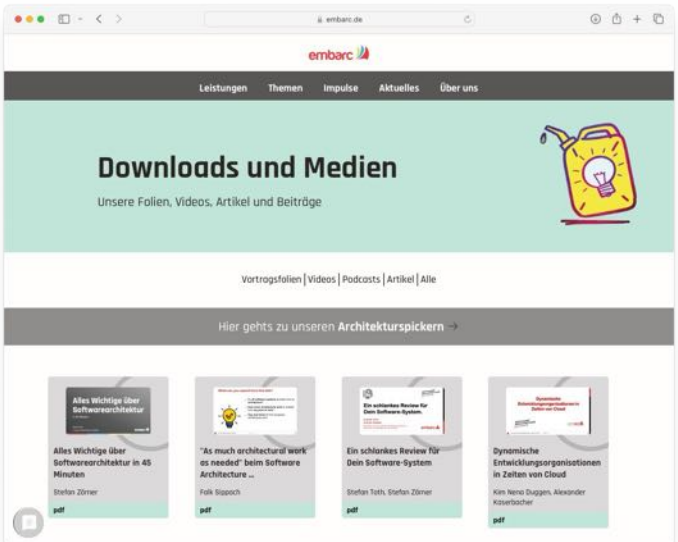
LASR skizziert in seinem Kern-Review erprobte Schritte für eben solch ein Review. Ihr könnt es **direkt in Eurem Team** in 2- 3 h „durchspielen". Bei Bedarf bietet LASR Vertiefungspunkte zur **Konfidenzerhöhung** an.

Artikel zum Einstieg in LASR (online)

Stefan Toth & Stefan Zörner:
„Leichtgewichtige Software-Reviews mit LASR“
Online, Informatik Aktuell

https://www.informatik-aktuell.de/entwicklung/methoden/leichtgewichtige-software-reviews-mit-lasr.html

Folien als PDF zum Download



➔ embarc.de/download/

Buchtipp zum Thema

Software-Systeme reviewen
mit dem Lightweight Approach
for Software Reviews - LASR

Leanpub

Autoren: Stefan Toth, Stefan Zörner
Verlag: Leanpub, September 2023
Sprache: Deutsch, EPUB, PDF

➔ leanpub.com/software-systeme-reviewen/

Rabatt-Code für Leanpub

Software-Systeme reviewen
mit dem Lightweight Approach
for Software Reviews - LASR

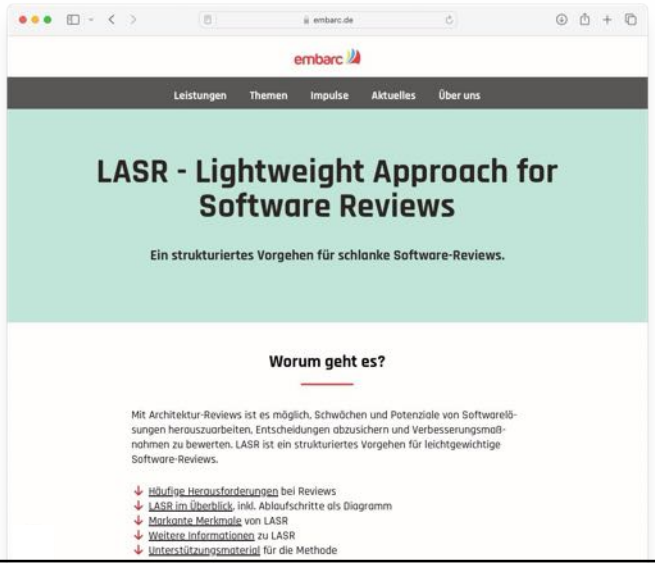
Für alle Teilnehmenden der
Berlin Expert Days, reduziert
den Preis um 45%

Coupon: JSD2024

SCAN ME

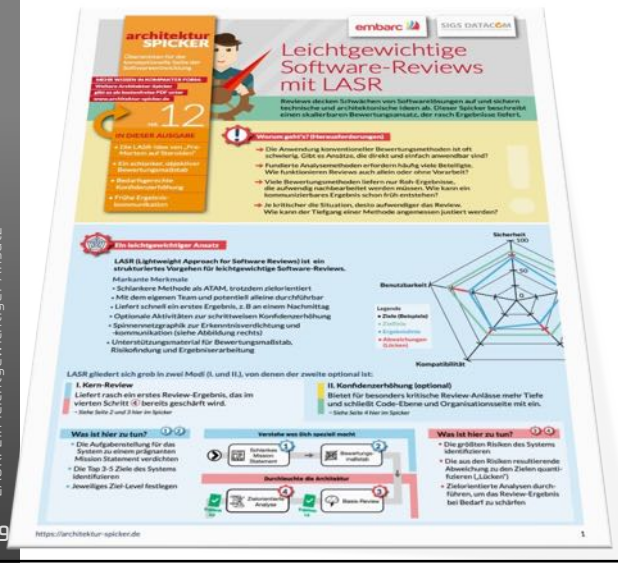
➔ leanpub.com/software-systeme-reviewen/c/JSD2024

LASR-Themenseite bei embarc



➔ embarc.de/lasr-reviews/

Architektur-Spicker zum Thema …



„Mit unseren Architektur-Spickern beleuchten wir die konzeptionelle Seite der Softwareentwicklung.“

Architektur-Spicker #12
Leichtgewichtige Software Reviews mit LASR

PDF, 4 Seiten
Kostenloser Download.

➔ architektur-spicker.de


Architektur-
Punsch
Unsere Speaker 2024
Alex Kaserbacher
Stefan Toth
Matthias Noab
Stefan Zörner
Anja Kammer
Michael Wyss
Kim Duggen
Felix Kammerlander
16. Dezember | online
Infos, Programm & Anmeldung
embarc.de/events/punsch-2024


embarc.de
LASR. Ein leichtgewichtiger Ansatz
91
Vielen Dank.
Ich freue mich auf Eure Fragen!
Stefan.Zoerner@embarc.de
linkedin.com/in/stefan-zoerner
@StefanZoerner@mastodon.social
embarc